Student Name: **Stud id:** **Sect#:** **#:**

University of Bahrain College of Information Technology
Department of Computer Science

ITCS332: Organization of Programming Languages **QUIZ#3: Chapter 16_LP**

****

1) In a given compound goal, the process of finding a complete proof for the first subgoal before working on others is called DEPTH-FIRST SEARCH; the process that works on all subgoals in parallel is called BREADTH-FIRST SEARCH;.

2) If the goal succeeded, control leaves through the *exit* port; if a goal failed, control leaves through the *fail* port.

3) A clausal form of propositions contains *conjunction* operators only in its right side and *disjunction* operators only in its left side.

4) The Prolog query: `?- X is 23 mod 8, Y is 20*2.5.` produces:

   X= 7 and *Y= 50.0*

```
formT(X,Y,R):- X>=Y, R is X+Y.
formT(X,Y,R):- X<Y, R is X-Y.
```

5) The Prolog query: `formT(40,25,U).` produces: *U = 65*

```
rat([],[]).
rat([H],[H]).
rat([H1,H2|T],[HN|TN]) :- H1 =< H2, HN is H1-H2,rat(T,TN).
rat([H1,H2|T],[HN|TN]) :- H1 > H2, HN is H1+H2,rat(T,TN).
```

6) The Prolog query: `?- rat([12,17,20,6,8,-8,-22],X).` produces:

   X = [-5, 26, 0, -22].

7) Write Prolog code named *repc* that accepts a list of numbers and replaces each negative value by −1, each positive value by +1, and keeps zeros without change. Sample run:

```
?- repc([-11,23,-9,0,8,-5],X).
X = [-1, 1, -1, 0, 1, -1] .

?- repc([-11,-23,0,0,0,-8,-5],X).
X = [-1, -1, 0, 0, 0, -1, -1]
```

```
repc([],[]).
repc([H|T],[HN|TN]):- H > 0, HN is +1, repc(T,TN).
repc([H|T],[HN|TN]):- H < 0, HN is -1, repc(T,TN).
repc([H|T],[HN|TN]):- H =:= 0, HN is H, repc(T,TN).
```

Student Name: **Stud id:** **Sect#:** **#:**

University of Bahrain College of Information Technology
Department of Computer Science

ITCS332: Organization of Programming Languages **QUIZ#3: Chapter 16_LP**

****

1) In Prolog, the unification of terms is performed by = operator and the calculation of terms is done by *is* operator.

2) There are 2 approaches of matching a given goal to facts in a database: FORWARD CHAINING (BOTTOM-UP RESOLUTION) and BACKWARD CHAINING (TOP-DOWN RESOLUTION).

3) The process of finding useful values for variables in propositions that allows matching process to succeed is called *unification*. The process of assigning temporary values to variables to allow unification to succeed is called *instantiation* .

4) The Prolog query: `?- X is 2**5, Y is 63 mod 11.` Produ`ces:

X= 32 and *Y= 8*

```
taz([],[]).
taz([H1,H2|T],[HR|TT]):- HR is H2*H1, taz(T,TT).
```

5) The Prolog query: `taz([5,3,11,2,9,-3],F).` produces: *F = [15, 22, -27].*

```
fat([],[]).
fat([H],[H]).
fat([H1,H2|T],[HN|TN]) :- H1 > H2, HN is H1-H2,fat(T,TN).
fat([H1,H2|T],[HN|TN]) :- H1 =< H2, HN is H1+H2,fat(T,TN).
```

6) The Prolog query: `fat([12,17,20,6,8,-8,-11],X).` produces:

X = [29, 14, 16, -11].

7) Write Prolog code named *pcat* that accepts a list of integers and replaces each odd value by −1, each even value by +1. Sample run:

```
?- pcat([-11,23,-9,0,8,-5],X).
X = [-1, -1, -1, 1, 1, -1] .

?- repc([12,17,20,6,8,-8,-5],X).
X = [1, 1, 1, 1, 1, -1, -1]
```

```
pcat([],[]).
pcat([H|T],[HN|TN]):- R is H mod 2,R =:=0, HN is +1, pcat(T,TN).
pcat([H|T],[HN|TN]):- R is H mod 2,R =\=0, HN is -1, pcat(T,TN).
```